CSC 1351: Final

VERSION A

Name: _____

1 Interfaces, Classes and Inheritance

2 Basic Data Types (arrays, lists, stacks, queues, trees, ...)

2.1

Does the following code compile? If it does not, how can it be fixed? If it does, what is it's output? Does it throw an exception? If so, how can it be fixed?

```
public class ArraysInJava
{
    public static void main(String[] args)
    {
        int[] i = new int[0];
        System.out.println(i[0]);
    }
}
```

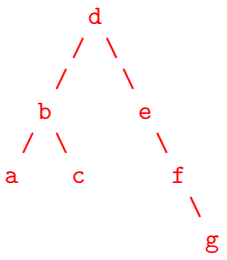
The code compiles, but when it runs it throws a ArrayIndexOutOfBoundsException

replace the code: int[] i = new int[0];

with the code: int[] i = new int[1];

2.2

The following values are added to a binary tree. Please draw the resulting tree structure. d b e f a c g



2.3

Consider the Btree below: Fix the add method. Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
import java.util.*;
class Node<T> {
  T data;
  Node<T> L, R;
  Node(T v) { data = v; } }
public class BTree<T extends Comparable<T>> {
  Node<T> head;
```

```
  public void add(T key) {
    if(head == null) head = new Node<>(key);
    else add(head,key); }
  private void add(Node<T> n,T key) {
    if(key.compareTo(n.data) > 0) {
    --------------------
      if(n.L == null) n.L = new Node<>(key);
      --------------------
      else add(n.L,key); }
      --------------------
    if(key.compareTo(key) < 0) {
    --------------------
      if(n.R == null) n.R = new Node<>(key);
      --------------------
      else add(n.R,key);
      --------------------
      } }
}
```

2.4

Find the best match between the column on the left and the column on the right.

(1) HashMap	5	(e) unique items
(2) ArrayList	7	(c) a subclass of Vector
(3) Queue	3	(d) an Interface
(4) TreeSet	1	(i) key values pairs maybe in order
(5) Set	2	(a) efficient item lookup
(6) LinkedList	9	(h) key value pairs traversed in order
(7) Stack	8	(g) requires a hashCode
(8) HashSet	6	(b) efficient insert and delete
(9) TreeMap	4	(f) requires a Comparator or Comparable

2.5

Which of the following values is the best hash code for String s?

1. s.charAt(0)
2. 1
3. s.length() ← This one

Answer 1 won't work if the string has zero length.
Answer 2 is usable, but highly inefficient.
Answer 3 will always work and provide some discrimination between items.

2.6

What is wrong with this code?

```
import java.util.*;
public class Monster implements Comparable<Monster> {
  int numTeeth, numClaws;
  String name;
  Monster(String n,int t,int c) { name = n; numTeeth = t; numClaws = 2; }
  public int compareTo(Monster that) {
    int diff = this.numTeeth - that.numTeeth;
    if(diff == 0) diff = this.numClaws - that.numClaws;
    return diff;
  }
}
```

The number of teeth and claws are not final, therefore sorts will be undone and binary tree lookups will fail.

2.7

What is wrong with this code?

```
import java.util.*;
public class MathWizard {
  final int level;
  int spells, id;
  MathWizard(int id,int lvl,int spells) {
    this.id = id; this.level = lvl; this.spells = spells;
  }
  public int hashCode() { return id; }
  public static void main(String[] args) {
    Set<MathWizard> s = new TreeSet<>();
    s.add(new MathWizard(1,2,3));
  }
}
```

MathWizard is not Comparable, which is what would be required for TreeSet. Alternatively, Set s could be initialized as a hash set. As it is, this code fails with a ClassCastException.

2.8

2.8.1

Write a method that, given an array of integers and an integer s, prints all pairs of array elements whose sum is equal to s. Don't print other pairs, print pairs only once, and print the smaller number first. Assume the array does not contain duplicate entries.

2.8.2

What is the big-O speed of your solution, if N is the size of the array?

3 Recursion

3.1

What is recursion?

1. Recursion is a generic class.
2. Recursion is a process of setting a value based on it's previous value.
3. Recursion is a process of defining a method that calls itself.
4. Recursion is a process of repeatedlty calling other methods.

Recursion is a process of defining a method that calls itself.

3.2

Which of these will happen if recursive method does not have a base case?

1. An infinite loop occurs, hanging forever.
2. The program stops after some time with an error.
3. After 1000000 calls it will be automatically stopped.
4. None of the mentioned

The program stops after some time with an error. (stack overflow)

3.3

What is the output of this program?

```
class recursion {
    int func (int n) {
        int result;
        result = func (n - 1);
        return result;
    }
}
public class Output {
    public static void main(String args[]) {
        recursion obj = new recursion() ;
        System.out.print(obj.func(12));
    }
}
```

1. 0
2. 1
3. 12
4. Compilation Error
5. Runtime Error

Runtime Error: Exception in thread main java.lang.StackOverflowError

3.4

Does the following code compile? If it does not, how can it be fixed? If it does, what is it's output? Does it throw an exception? If so, how can it be fixed?

```
public class Series {
  public static int func(int j) {
  System.out.println(j);
  if (j==1) return 1;
    return 2*func(j-1) + 5*func(j-2);
  }
  public static void main(String[] args) {
    int N = Integer.parseInt(args[0]);
    if (N<1) {
      System.out.println("invalid␣argument");
      return;
    }
    System.out.println(func(N));
  }
}
```

The code compiles, but when it runs it throws a StackOverflowError

replace the code: if (j==1) return 1;

with the code: if (j<1) return 1;

4 Performance

4.1

The following table gives approximate running times for a program with N inputs for various values of N.

N		time
1000	5	seconds
2000	20	seconds
5000	2	minutes
10000	8	minutes

Which of the following best describes the likely running time of this program for N = 100, 000?

1. A few minutes
2. A few hours
3. Half a day
4. A few days

The scaling seems to be approximately N^2, so the estimated runtime would be $10^2 * 8$ minutes = 800 minutes: about half a day.

5 Files and Exceptions

6 Generic Classes

6.1

The following code dies with a ClassCastException on the line indicated whenever it is used to permute a list with more than one element. What's wrong? How would generics have solved the problem?

```
import java.util.ArrayList;

public class Permute {
    // Return a list of lists, where each list in the
    // return list is a different permutation of the input list.
    public static ArrayList permutations(ArrayList list) {
        ArrayList permuted_list = new ArrayList();

        if(list.size() < 2)
          return list;

        for(int i=0;i<list.size();i++) {
            ArrayList shorter_list = new ArrayList();
            for(int j=0;j<list.size();j++) {
                if(i != j)
                    shorter_list.add(list.get(j));
            }
            for(Object o : permutations(shorter_list)) {
                // ClassCastException occurs on this line
                ArrayList ilist = (ArrayList)o;
                ilist.add(list.get(i));
                permuted_list.add(ilist);
            }
        }
        return permuted_list;
    }
}
```

The problem is what happens when the list size is less than 2. The code above returns a list instead of a list of lists. Generic code prevents the problem by producing a compile error when the programmer attempts to return a list of the wrong type.

6.2

Does the following code compile? If it does not, how can it be fixed? If it does, what is it's output? Does it throw an exception? If so, how can it be fixed?

```
import java.util.*;
public class Coord implements Comparable<Coord> {
  final int x,y;
  public Coord(int x,int y) { this.x = x; this.y = y; }
  @Override
    public int compareTo(Object o) { Coord that = (Coord)o;
    return 10000*(this.x - that.x) + (this.y - that.y);
  }
  public static void main(String[] args) {
    List<Coord> li = new ArrayList<>();
    li.add(new Coord(3,4));
    li.add(new Coord(9,2));
    Collections.sort(li);
  }
}
```

The code does not compile.

replace the code: public int compareTo(Object o) { Coord that = (Coord)o;

with the code: public int compareTo(Coord that) {